Programming Problems Set

Analysis and Design of Algorithms

April 4, 2019

793 Network Connections

Bob, who is a network administrator, supervises a network of computers. He is keeping a log connections between the computers in the network. Each connection is bi–directional. Two computers are interconnected if they are directly connected or if they are interconnected with the same computer. Occasionally, Bob has to decide, quickly, whether two given computers are connected, directly or indirectly, according to the log information.

Write a program which based on information input from a text file counts the number of successful and the number of unsuccessful answers to the questions of the kind:

is $computer_i$ interconnected with $computer_j$?

Input

The input begins with a single positive integer on a line by itself, indicating the number of the cases following. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

For each test case, the input must follow the description below.

1. The number of computers in the network (a strictly positive integer);
2. A list of pairs of the form:
   (a) `c` $computer_i$ $computer_j$, where $computer_i$ and $computer_j$ are integers from 1 to $no\_of\_computers$. A pair of this form shows that $computer_i$ and $computer_j$ **get interconnected**.
   (b) `q` $computer_i$ $computer_j$, where $computer_i$ and $computer_j$ are integers from 1 to $no\_of\_computers$. A pair of this form stands for the question:
   is $computer_i$ **interconnected** with $computer_j$?

   Each pair is on a separate line. Pairs can appear in any order, regardless of their type. The log is updated after each pair of type (a) and each pair of type (b) is processed according to the current network configuration.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The program prints two integer numbers to the standard output on the same line, in the order: '$successful\ answers$, $unsuccessful\ answers$', as shown in the sample output.

Note:

For example, the first input illustrated in the sample below corresponds to a network of 10 computers and 7 pairs. There are '1' successfully answered questions and '2' unsuccessfully answered questions.

Sample Input

```
2

10
c 1 5
c 2 7
q 7 1
c 3 9
q 9 6
c 2 5
q 7 5

1
q 1 1
c 1 1
q 1 1
```

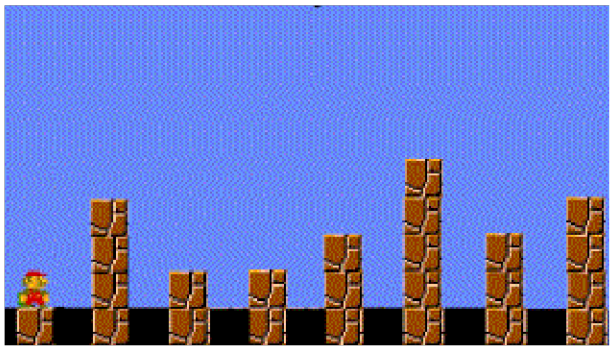
Sample Input

```
1,2

2,0
```

UVa Online Judge

11764 Jumping Mario



Mario is in the final castle. He now needs to jump over few walls and then enter the Koopa's Chamber where he has to defeat the monster in order to save the princess. For this problem, we are only concerned with the "jumping over the wall" part. You will be given the heights of N walls from left to right. Mario is currently standing on the first wall. He has to jump to the adjacent walls one after another until he reaches the last one. That means, he will make $(N-1)$ jumps. A *high jump* is one where Mario has to jump to a taller wall, and similarly, a low jump is one where Mario has to jump to a shorter wall. Can you find out the total number of *high jumps* and *low jumps* Mario has to make?

Input

The first line of input is an integer T ($T < 30$) that indicates the number of test cases. Each case starts with an integer N ($0 < N < 50$) that determines the number of walls. The next line gives the height of the N walls from left to right. Each height is a positive integer not exceeding 10.

Output

For each case, output the case number followed by 2 integers, total high jumps and total low jumps, respectively. Look at the sample for exact format.

Sample Input

```
3
8
1 4 2 2 3 5 3 4
1
9
5
1 2 3 4 5
```

Sample Output

```
Case 1: 4 2
Case 2: 0 0
Case 3: 4 0
```

102 Ecological Bin Packing

Bin packing, or the placement of objects of certain weights into different bins subject to certain constraints, is an historically interesting problem. Some bin packing problems are NP-complete but are amenable to dynamic programming solutions or to approximately optimal heuristic solutions.

In this problem you will be solving a bin packing problem that deals with recycling glass.

Recycling glass requires that the glass be separated by color into one of three categories: brown glass, green glass, and clear glass. In this problem you will be given three recycling bins, each containing a specified number of brown, green and clear bottles. In order to be recycled, the bottles will need to be moved so that each bin contains bottles of only one color.

The problem is to minimize the number of bottles that are moved. You may assume that the only problem is to minimize the number of movements between boxes.

For the purposes of this problem, each bin has infinite capacity and the only constraint is moving the bottles so that each bin contains bottles of a single color. The total number of bottles will never exceed 2^{31}.

Input

The input consists of a series of lines with each line containing 9 integers. The first three integers on a line represent the number of brown, green, and clear bottles (respectively) in bin number 1, the second three represent the number of brown, green and clear bottles (respectively) in bin number 2, and the last three integers represent the number of brown, green, and clear bottles (respectively) in bin number 3. For example, the line

10 15 20 30 12 8 15 8 31

indicates that there are 20 clear bottles in bin 1, 12 green bottles in bin 2, and 15 brown bottles in bin 3.

Integers on a line will be separated by one or more spaces. Your program should process all lines in the input file.

Output

For each line of input there will be one line of output indicating what color bottles go in what bin to minimize the number of bottle movements. You should also print the minimum number of bottle movements.

The output should consist of a string of the three upper case characters 'G', 'B', 'C' (representing the colors green, brown, and clear) representing the color associated with each bin.

The first character of the string represents the color associated with the first bin, the second character of the string represents the color associated with the second bin, and the third character represents the color associated with the third bin.

The integer indicating the minimum number of bottle movements should follow the string.

If more than one order of brown, green, and clear bins yields the minimum number of movements then the alphabetically first string representing a minimal configuration should be printed.

Sample Input

```
1 2 3 4 5 6 7 8 9
5 10 5 20 10 5 10 20 10
```

Sample Output

```
BCG 30
CBG 50
```

UVa Online Judge

156 Ananagrams

Most crossword puzzle fans are used to *anagrams* — groups of words with the same letters in different orders — for example OPTS, SPOT, STOP, POTS and POST. Some words however do not have this attribute, no matter how you rearrange their letters, you cannot form another word. Such words are called *ananagrams*, an example is QUIZ.

Obviously such definitions depend on the domain within which we are working; you might think that ATHENE is an ananagram, whereas any chemist would quickly produce ETHANE. One possible domain would be the entire English language, but this could lead to some problems. One could restrict the domain to, say, Music, in which case SCALE becomes a *relative ananagram* (LACES is not in the same domain) but NOTE is not since it can produce TONE.

Write a program that will read in the dictionary of a restricted domain and determine the relative ananagrams. Note that single letter words are, ipso facto, relative ananagrams since they cannot be "rearranged" at all. The dictionary will contain no more than 1000 words.

Input

Input will consist of a series of lines. No line will be more than 80 characters long, but may contain any number of words. Words consist of up to 20 upper and/or lower case letters, and will not be broken across lines. Spaces may appear freely around words, and at least one space separates multiple words on the same line. Note that words that contain the same letters but of differing case are considered to be anagrams of each other, thus '`tIeD`' and '`EdiT`' are anagrams. The file will be terminated by a line consisting of a single '`#`'.

Output

Output will consist of a series of lines. Each line will consist of a single word that is a relative ananagram in the input dictionary. Words must be output in lexicographic (case-sensitive) order. There will always be at least one relative ananagram.

Sample Input

```
ladder came tape soon leader acme RIDE lone Dreis peat
 ScAlE orb  eye  Rides dealer  NotE derail LaCeS  drIed
noel dire Disk mace Rob dries
#
```

Sample Output

```
Disk
NotE
derail
drIed
eye
ladder
soon
```

UVa Online Judge

10420 List of Conquests

In Act I, Leporello is telling Donna Elvira about his master's long list of conquests:

> "This is the list of the beauties my master has loved, a list I've made out myself: take a look, read it with me. In Italy six hundred and forty, in Germany two hundred and thirty-one, a hundred in France, ninety-one in Turkey; but in Spain already a thousand and three! Among them are country girls, waiting-maids, city beauties; there are countesses, baronesses, marchionesses, princesses: women of every rank, of every size, of every age." (Madamina, il catalogo questo)

As Leporello records all the "beauties" Don Giovanni "loved" in chronological order, it is very troublesome for him to present his master's conquest to others because he needs to count the number of "beauties" by their nationality each time. You are to help Leporello to count.

Input

The input consists of at most 2000 lines. The first line contains a number n, indicating that there will be n more lines. Each following line, with at most 75 characters, contains a country (the first word) and the name of a woman (the rest of the words in the line) Giovanni loved. You may assume that the name of all countries consist of only one word.

Output

The output consists of lines in alphabetical order. Each line starts with the name of a country, followed by the total number of women Giovanni loved in that country, separated by a space.

Sample Input

```
3
Spain Donna Elvira
England Jane Doe
Spain Donna Anna
```

Sample Output

```
England 1
Spain 2
```

UVa Online Judge

12356 Army buddies

Nlogonia is fighting a ruthless war against the neighboring country of Cubiconia. The Chief General of Nlogonia's Army decided to attack the enemy with a linear formation of soldiers, that would advance together until conquering the neighboring country. Before the battle, the Chief General ordered that each soldier in the attack line, besides protecting himself and attacking, should also protect his two (nearest) neighbors in the line, one to his left and one to his right. The Chief General told the soldiers that for each of them, his "buddies" would be these two neighbors, if such neighbors existed (because the leftmost soldier does not have a left neighbor and the rightmost soldier does not have a right neighbor). The Chief General also told the soldiers that protecting their buddies was very important to prevent the attack line from being broken. So important that, if the left or right buddy of a soldier is killed, then the next living neighbor to the left or to the right of the soldier, respectively, should become his buddy.

The battle is fierce, and many soldiers in the attack line are being killed by fire shots, grenades and bombs. But following the Chief General's orders, immediately after knowing about losses in the attack line, the Army's information systems division has to inform the soldiers who their new buddies are.

You are given the number of soldiers in the attack line, and a sequence of loss reports. Each loss report describes a group of contiguous soldiers in the attack line that were just killed in the battle. Write a program that, for each loss report, prints the new buddies formed.

Input

Each test case is described using several lines. The first input line contains two integers S and B representing respectively the number of soldiers in the attack line, and the number of loss reports ($1 \leq B \leq S \leq 10^5$). Soldiers are identified by different integers from 1 to S, according to their positions in the attack line, being 1 the leftmost soldier and S the rightmost soldier. Each of the next B input lines describes a loss report using two integers L (left) and R (right), meaning that soldiers from L to R were killed ($1 \leq L \leq R \leq S$). You may assume that until that moment those soldiers were alive and were just killed.

The last test case is followed by a line containing two zeros.

Output

For each test case output $B+1$ lines. In the i-th output line write the new buddies formed by removing from the attack line the soldiers that were just killed according to the i-th loss report. That is, for the loss report 'L R', print the first surviving soldier to the left of L, and the first surviving soldier to the right of R. For each direction, print the character '*' (asterisk) if there is no surviving soldier in that direction. Print a line containing a single character '-' (hyphen) after each test case.

Sample Input

```
1 1
1 1
10 4
2 5
6 9
1 1
10 10
```

```
5 1
1 1
0 0
```

Sample Output

```
* *
-
1 6
1 10
* 10
* *
-
* 2
-
```

UVa Online Judge

10077 The Stern-Brocot Number System

The *Stern-Brocot tree* is a beautiful way for constructing the set of all nonnegative fractions $\frac{m}{n}$ where m and n are relatively prime. The idea is to start with two fractions $\left(\frac{0}{1}, \frac{1}{0}\right)$ and then repeat the following operations as many times as desired:

Insert $\frac{m+m'}{n+n'}$ between two adjacent fractions $\frac{m}{n}$ and $\frac{m'}{n'}$.

For example, the first step gives us one new entry between $\frac{0}{1}$ and $\frac{1}{0}$,

$$\frac{0}{1}, \frac{1}{1}, \frac{1}{0};$$

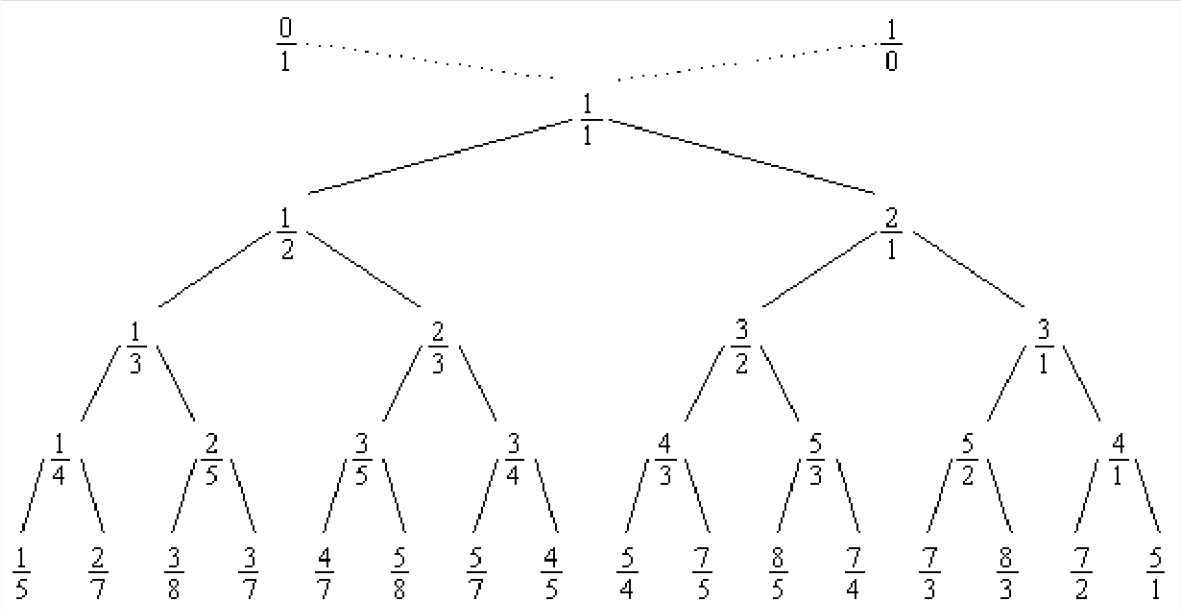
and the next gives two more:

$$\frac{0}{1}, \frac{1}{2}, \frac{1}{1}, \frac{2}{1}, \frac{1}{0}.$$

The next gives four more,

$$\frac{0}{1}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{1}{1}, \frac{3}{2}, \frac{2}{1}, \frac{3}{1}, \frac{1}{0};$$

and then we will get 8, 16, and so on. The entire array can be regarded as an infinite binary tree structure whose top levels look like this:



The construction preserves order, and we couldn't possibly get the same fraction in two different places.

We can, in fact, regard the *Stern-Brocot tree* as a *number system* for representing rational numbers, because each positive, reduced fraction occurs exactly once. Let's use the letters 'L' and 'R' to stand for going down to the left or right branch as we proceed from the root of the tree to a particular fraction; then a string of L's and R's uniquely identifies a place in the tree. For example, LRRL means that we go left from $\frac{1}{1}$ down to $\frac{1}{2}$, then right to $\frac{2}{3}$, then right to $\frac{3}{4}$, then left to $\frac{5}{7}$. We can consider LRRL to be a representation of $\frac{5}{7}$. Every positive fraction gets represented in this way as a unique string of L's and R's.

Well, actually there's a slight problem: The fraction $\frac{1}{1}$ corresponds to the empty string, and we need a notation for that. Let's agree to call it I, because that looks something like 1 and it stands for "identity".

In this problem, given a positive rational fraction, you are expected to represent it in *Stern-Brocot number system*.

Input

The input file contains multiple test cases. Each test case consists of a line contains two positive integers m and n where m and n are relatively prime. The input terminates with a test case containing two `1`'s for m and n, and this case must not be processed.

Output

For each test case in the input file output a line containing the representation of the given fraction in the *Stern-Brocot number system*.

Sample Input

```
5 7
878 323
1 1
```

Sample Output

```
LRRL
RRLRRLRLLLLRLRRR
```

UVa Online Judge

10945 Mother Bear

Unforunately for our lazy "heroes", the nuts were planted by an evil bear known as.. **Dave**, and they've fallen right into his trap. Dave is not just any bear, he's a talking bear, but he can only understand sentences that are palindromes. While Larry was dazed and confused, Ryan figured this out, but need a way to make sure his sentences are palindromic. So he pulled out his trustly iPod, which thankfully have this program you wrote just for this purpose... or did you?

Input

You'll be given many sentences. You have to determine if they are palindromes or not, ignoring case and punctuations. Every sentence will only contain the letters `A-Z`, `a-z`, '`.`', '`,`', '`!`', '`?`'. The end of input will be a line containing the word '`DONE`', which should not be processed.

Output

On each input, output '`You won't be eaten!`' if it is a palindrome, and '`Uh oh..`' if it is not a palindrome.

Sample Input

```
Madam, Im adam!
Roma tibi subito motibus ibit amor.
Me so hungry!
Si nummi immunis
DONE
```

Sample Output

```
You won't be eaten!
You won't be eaten!
Uh oh..
You won't be eaten!
```

UVa Online Judge

732 Anagrams by Stack

How can anagrams result from sequences of stack operations? There are two sequences of stack operators which can convert TROT to TORT:

```
[
i i i i o o o o
i o i i o o i o
]
```

where 'i' stands for Push and 'o' stands for Pop. Your program should, given pairs of words produce sequences of stack operations which convert the first word to the second.

Input

The input will consist of several lines of input. The first line of each pair of input lines is to be considered as a source word (which does not include the end-of-line character). The second line (again, not including the end-of-line character) of each pair is a target word.

Output

For each input pair, your program should produce a sorted list of valid sequences of 'i' and 'o' which produce the target word from the source word. Each list should be delimited by

```
[
]
```

and the sequences should be printed in "dictionary order". Within each sequence, each 'i' and 'o' is followed by a single space and each sequence is terminated by a new line.

Process

A stack is a data storage and retrieval structure permitting two operations:

Push — to insert an item and

Pop — to retrieve the most recently pushed item

We will use the symbol 'i' (in) for push and 'o' (out) for pop operations for an initially empty stack of characters. Given an input word, some sequences of push and pop operations are valid in that every character of the word is both pushed and popped, and furthermore, no attempt is ever made to pop the empty stack. For example, if the word FOO is input, then the sequence:

`i i o i o o` is valid, but
`i i o` is not (it's too short), neither is
`i i o o o i` (there's an illegal pop of an empty stack)

Valid sequences yield rearrangements of the letters in an input word. For example, the input word FOO and the sequence 'i i o i o o' produce the anagram OOF. So also would the sequence 'i i i o o o'. You are to write a program to input pairs of words and output all the valid sequences of 'i' and 'o' which will produce the second member of each pair from the first.

Sample Input

```
madam
adamm
bahama
bahama
long
short
eric
rice
```

Sample Output

```
[
i i i i o o o i o o
i i i i o o o o i o
i i o i o i o i o o
i i o i o i o o i o
]
[
i o i i i o o i i o o o
i o i i i o o o i o i o
i o i o i o i i i o o o
i o i o i o i o i o i o
]
[
]
[
i i o i o i o o
]
```

UVa Online Judge

10071 Back to High School Physics

A particle has initial velocity and acceleration. If its velocity after certain time is v then what will its displacement be in twice of that time?

Input

The input will contain two integers in each line. Each line makes one set of input. These two integers denote the value of v ($-100 \leq v \leq 100$) and t ($0 \leq t \leq 200$) (t means at the time the particle gains that velocity)

Output

For each line of input print a single integer in one line denoting the displacement in double of that time.

Sample Input

```
0 0
5 12
```

Sample Output

```
0
120
```

1208 Oreon

In the 25th century, civilization is struck by a series of calamities that eventually led mankind to build walled cities interconnected by tunnel bridges to facilitate transportation. Each walled city possesses a unique ore required to build and repair all infrastructure including the tunnels. This material which when combined with other ores from all others cities form an almost indestructible material called "oreon".

Outside the walled cities are uncivilized barbarians armed with antiquated but destructive weaponry which can effectively shoot down any air transport, but only damage and not penetrate tunnel bridges. Thus each city is interconnected to more than one city in order to have access redundancy in case one of its interconnecting tunnels is damaged.

If a tunnel is damaged, it becomes impassable and would require a substantial amount of "oreon" to repair the damage. When a single city is made isolated, meaning all of its interconnections are damaged, "oreon" cannot be manufactured which may lead to the eventual destruction of the wall fortifying the city. You, being the head of the homeland defense unit, are tasked to ensure that all cities remain accessible even by at least a single interconnecting tunnel at all times. Faced with only a limited manpower in the defense unit, you have to determine which tunnel to protect using the least number of people and ensure that no city will be isolated.

Figure 2 shows a map of the walled cities, their interconnecting tunnels and the number of security personnel.

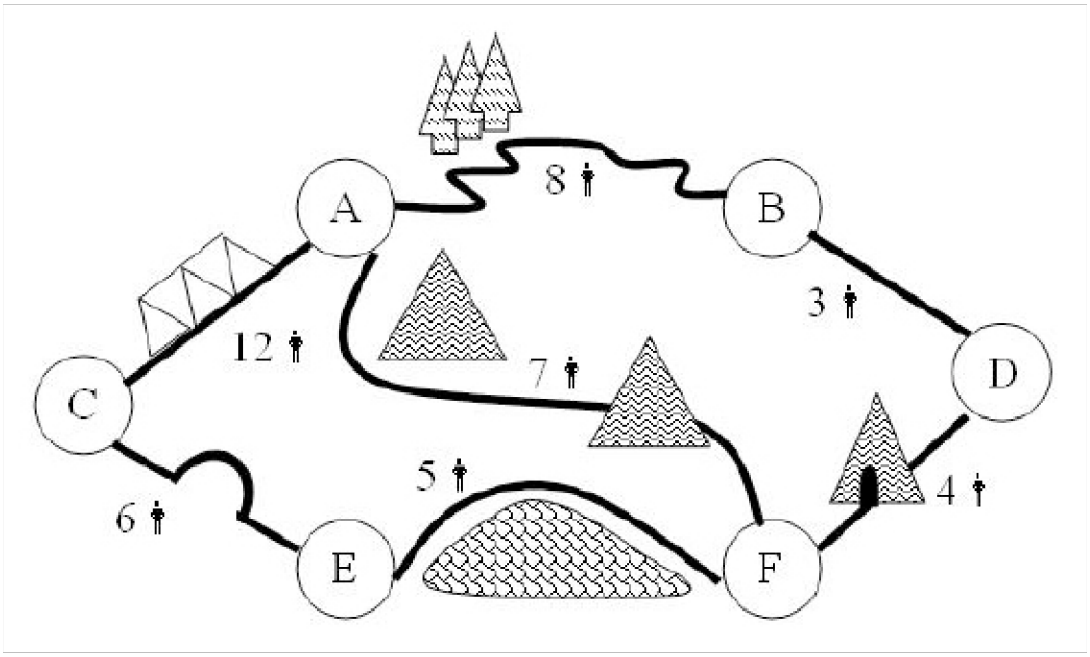


Figure 2: Map of six cities and its interconnecting tunnels

Input

The input will contain several test cases. The first line will indicate the number of test cases. Each test case begins with a number representing the number of walled cities. Cities are labeled alphabetically using the letters in the English alphabet. The subsequent lines contain the number of security personnel needed to protect the tunnel connecting each city to all other cities. A value of zero implies no security

personnel needed since no tunnel exists. You are to output which tunnel should be protected and how many personnel are needed for each tunnel.

Output

The output shows the tunnel connection which is named after the cities that it connects and the number of personel needed to protect the tunnel.

Sample Input

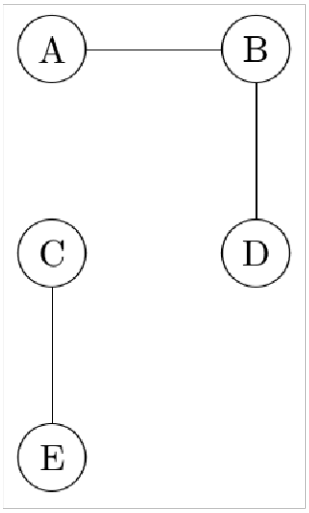
```
1
6
0, 8, 12, 0, 0, 7
8, 0, 0, 3, 0, 0
12, 0, 0, 0, 6, 0
0, 3, 0, 0, 0, 4
0, 0, 6, 0, 0, 5
7, 0, 0, 4, 5, 0
```

Sample Output

```
Case 1:
B-D 3
D-F 4
E-F 5
C-E 6
A-F 7
```

UVa Online Judge

459 Graph Connectivity



Consider a graph G formed from a large number of nodes connected by edges. G is said to be *connected* if a path can be found in 0 or more steps between any pair of nodes in G. For example, the graph below is not connected because there is no path from A to C.

This graph contains, however, a number of subgraphs that are connected, one for each of the following sets of nodes: {A}, {B}, {C}, {D}, {E}, {A,B}, {B,D}, {C,E}, {A,B,D}

A connected subgraph is *maximal* if there are no nodes and edges in the original graph that could be added to the subgraph and still leave it connected. There are two maximal connected subgraphs above, one associated with the nodes $\{A, B, D\}$ and the other with the nodes $\{C, E\}$.

Write a program to determine the number of maximal connected subgraphs of a given graph.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of each input set contains a single uppercase alphabetic character. This character represents the largest node name in the graph. Each successive line contains a pair of uppercase alphabetic characters denoting an edge in the graph.

The sample input section contains a possible input set for the graph pictured above.

Input is terminated by a blank line.

Output

For each test case, write in the output the number of maximal connected subgraphs. The outputs of two consecutive cases will be separated by a blank line.

Sample Input

```
1

E
AB
CE
DB
EC
```

Sample Output

```
2
```

UVa Online Judge

872 Ordering

Order is an important concept in mathematics and in computer science. For example, Zorns Lemma states: *a partially ordered set in which every chain has an upper bound contains a maximal element.* Order is also important in reasoning about the fix-point semantics of programs.

This problem involves neither Zorns Lemma nor fix-point semantics, but does involve order.

Given a list of variable constraints of the form $A < B$, you are to write a program that prints all orderings of the variables that are consistent with the constraints. For example, given the constraints $A < B$ and $A < C$ there are two orderings of the variables A, B, and C that are consistent with these constraints: ABC and ACB.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The input consists of two lines: a list of variables on one line followed by a list of constraints of the form $A < B$ on the next line. Variables and constraints are separated by single spaces.

All variables are single character, upper-case letters. There will be at least two variables, and no more than 20 variables. There will be at least one constraint, and no more than 50 constraints. There will be no more than 300 orderings consistent with the constraints in a specification.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

All orderings consistent with the constraints should be printed. Orderings are printed in alphabetical order, one per line. Characters on a line are separated by a space. If no ordering is possible the output is a single line with the word 'NO'.

Sample Input

```
1

A B F G
A<B B<F
```

Sample Output

```
A B F G
A B G F
A G B F
G A B F
```